

TIC-TAC-TOE USING AI ALGORITHM

Submitted by: Nakul Malhotra

Roll No. 2019-IMG-039

Abstract:

In this project AI-algorithm (Minimax algorithm) is used to form tic-tac-toe game which use the optimal path to play the game.

Introduction and Methodology:

- Minimax algorithm:

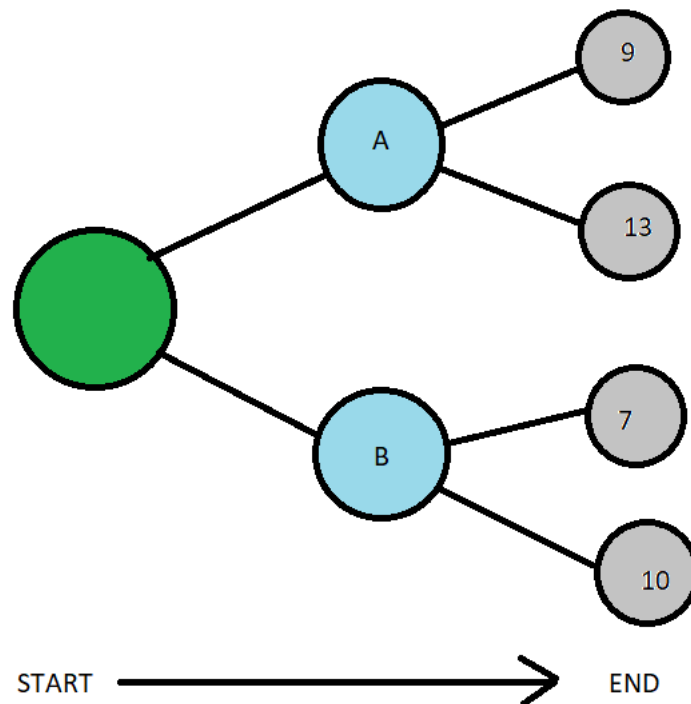
Minimax algorithm is a kind of backtracking algorithm which finds the optimal path based on the estimation that the opponent also chooses the optimal path. This algorithm is used in various games like tic-tac-toe, chess etc.

In this we use minimiser and maximiser as sort of 2 players which plays the role as suggested by their name. By using heuristic approach, we can calculate whether the value is positive (favours maximiser) or negative (favours minimiser).

In general, minimax algorithm backtracks each possibility and find the optimal solution. For example if the algorithm is made to favour the minimiser, then it will search the possibility so the final output will be as minimum as possible.

- Example and relation to the game

Let's say for example, possible paths for a short game is like the figure shown in the tree below-




If maximiser gets the first chance, we assume it will choose optimal path (A in this case to get 13) then according to minimax theory the next move will be towards 9. If the minimizer gets the chance to start the game then it will backtrack all the possible situation (i.e., start->A->9, start->A->13, start->B->7 and start->b->10) and choose the path where it can minimize the output hence it will choose B.

Same concept applies when we use this algorithm in tic-tac-toe game. Based on the condition of the board the algo will determine the optimal path to end the game. So it will provide the best place to make the next move.


Result:

The following pictures are the output of a sample match against the AI.

 C:\windows\py.exe

Choose X or O

Chosen: ☐

 C:\windows\py.exe

First to start?[y/n]: y

```
C:\windows\py.exe
Human turn [X]

-----
|   ||   ||   |
-----
|   ||   ||   |
-----
|   ||   ||   |
-----
Use numpad (1..9):
```

```
C:\windows\py.exe
Human turn [X]

-----
| o ||   ||   |
-----
|   || x ||   |
-----
|   ||   ||   |
-----
Use numpad (1..9):
```

```
C:\windows\py.exe
Computer turn [O]

-----
| o | |   | |   |
-----
|   | | x | | x |
-----
|   | |   | |   |
-----
```

```
C:\windows\py.exe
Human turn [X]

-----
| o | |   | |   |
-----
| o | | x | | x |
-----
|   | |   | |   |
-----
Use numpad (1..9): 7
```

```
C:\windows\py.exe
Human turn [X]

-----
| o ||   || o |
-----
| o || x || x |
-----
| x ||   ||   |
-----

Use numpad (1..9): 2
```

```
C:\windows\py.exe
Human turn [X]

-----
| o || x || o |
-----
| o || x || x |
-----
| x || o ||   |
-----

Use numpad (1..9):
```

See in this game both human and computer makes every possible move to victory and if both human and computer choose the optimal path the game ends in a tie (as you can see in the last pic). So in short, the AI algo we used always chooses the optimal solution which results in either victory of computer (if the player doesn't choose the optimal path himself) or in tie (if player chooses the optimal path).

Conclusion:

We have made tic-tac-toe game with help of AI so it will become unbeatable.

It proves we can use AI algorithms to make computer apply the logic which humans use in the game.